RZ/A2M Group

MIPI Driver

Introduction

This application note describes about the functional specification of MIPI and Video Input Module(VIN) Driver for RZ/AM.

Target Device

RZ/A2M

Contents

1. Specifications

This driver uses the RZ/A2M group on-chip MIPI CSI2 Interface (here after MIPI) and Video Input Module (here after VIN) to capture the image data and transfer to the memory.

Table 1-1 shows the peripheral functions to be used by MIPI driver.

Table 1-1 Peripheral functions to be used by MIPI driver

Classification	Item	Implemented Function	Description	Remarks
Camera	Data transfer	Data lane swapping	2 lanes or 1 lane processing	
		Transfer rate	80MHz to 1GHz	
	Data correction	Packet header	ECC 1bit error correction 2bit or more error detection	
		Payload data	CRC error detection	
	Input format	RAW 8bit	Bayer or gray scale	
	Capture pixel size	Optional	MAX 2048 pixels x 2048 lines	
	Clipping size	Optional	MAX 2048 pixels x 2048 lines	
	Capture mode	Single / Continuous switching	Single frame or continuous frame	
		Field	Odd-field capture mode Even- / odd-field capture mode Even-field capture mode	
Memory write	Output format	RAW 8bit	Bayer or gray scale	

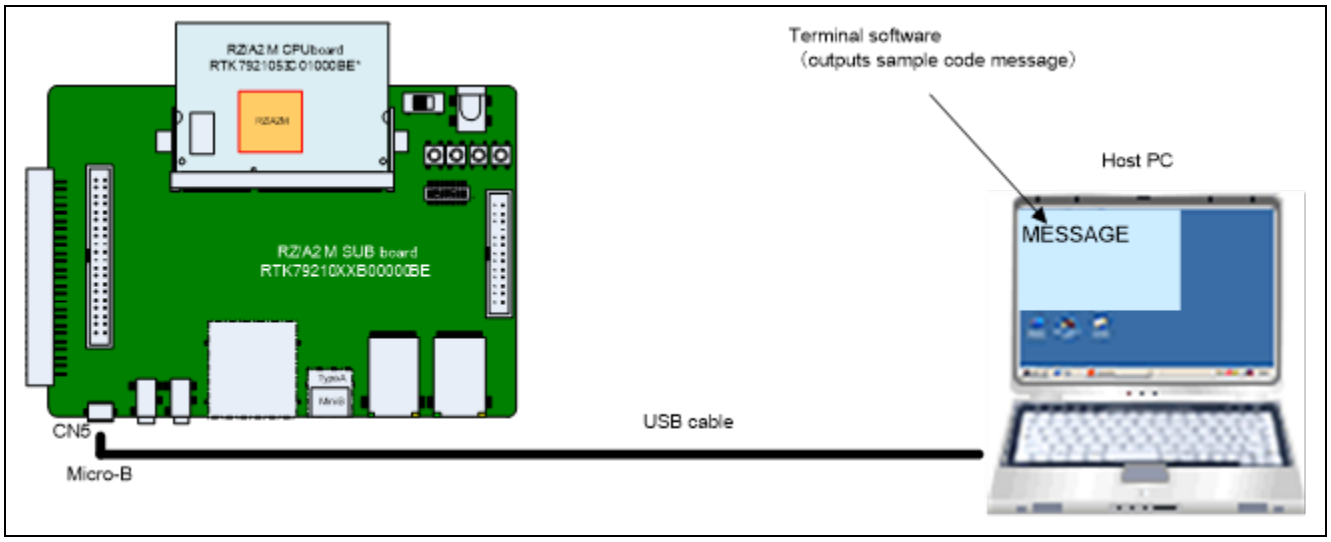

Figure 1.1 Operation check conditions

2. Operation Confirmation Conditions

The sample code of this application note supports following environment.

Table 2.1 Peripheral device used(1/2)

Peripheral device	Usage
MCU Used	RZ/A2M
Operating frequency[MHz] (Note)	CPU Clock (Iφ) : 528MHz Image processing clock (Gφ) : 264MHz Internal Bus Clock (Bφ) : 132MHz Peripheral Clock 1 (P1φ) : 66MHz Peripheral Clock 0 (P0φ) : 33MHz QSPI0_SPCLK : 66MHz CKIO : 132MHz
Operating voltage	Power supply voltage (I/O): 3.3 V Power supply voltage (either 1.8V or 3.3V I/O (PVcc SPI)) : 3.3V Power supply voltage (internal): 1.2 V
Integrated development environment	e2 studio V7.3.0
C compiler	“GNU Arm Embedded Tool chain 6-2017-q2-update” compiler options(except directory path) Release: -mcpu=cortex-a9 -march=armv7-a -marm -mlittle-endian -mfloat-abi=hard -mfpu=neon -mno-unaligned-access -Os -ffunction-sections -fdata-sections -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wpadded -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -Wnull-dereference -Wmaybe-uninitialized -Wstack-usage=100 -fabi-version=0 Hardware Debug: -mcpu=cortex-a9 -march=armv7-a -marm -mlittle-endian -mfloat-abi=hard -mfpu=neon -mno-unaligned-access -Og -ffunction-sections -fdata-sections -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wpadded -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -Wnull-dereference -Wmaybe-uninitialized -g3 -Wstack-usage=100 -fabi-version=0

Note: The operating frequency used in clock mode 1 (Clock input of 24MHz from EXTAL pin)

Table 2.2 Peripheral device used(2/2)

Operation mode	Boot mode 3 (Serial Flash boot 3.3V)
Terminal software communication settings	Communication speed: 115200bps Data length: 8 bits Parity: None Stop bits: 1 bit Flow control: None
Board to be used	RZ/A2M CPU board RTK7921053C00000BE RZ/A2M SUB board RTK79210XXB00000BE
Device (functionality to be used on the board)	Serial flash memory allocated to SPI multi-I/O bus space (channel 0) Manufacturer : Macronix Inc. Model Name : MX25L51245GXD RL78/G1C (This device communications the host PC by convert USB Communication and Serial Communication.) LED1

3. Hardware Description

3.1 List of pins to be used

Table 3-1 lists the pins to be used and describes their functionalities.

Table 3-1 List of pins to be used

Pin name	I/O	Description	Target board connection
CSI_DATA0P	Input	Differential positive receiving data input on CSI2 lane 0	Designated pin
CSI_DATA0N	Input	Differential negative receiving data input on CSI2 lane 0	Designated pin
CSI_DATA1P	Input	Differential positive receiving data input on CSI2 lane 1	Designated pin
CSI_DATA1N	Input	Differential negative receiving data input on CSI2 lane 1	Designated pin
CSI_CLKP	Input	Differential positive reception input on CSI2 clock lane	Designated pin
CSI_CLKN	Input	Differential negative reception input on CSI2 clock lane	Designated pin

4. Software Description

4.1 Files

Table 4-1 shows the files used by MIPI driver.

Table 4-1 Files used by MIPI driver

File name	Description
r_mipi_api.c	MIPI driver API functions Source file which is wrote API functions of MIPI driver
r_mipi_api.h	MIPI driver API definitions Header file which defines API functions prototype of MIPI driver and several parameters for MIPI driver
r_mipi_userdef_api.c	MIPI driver user definition functions Source file which depends on user environment part such as interrupt settings for operate MIPI driver and so on
r_mipi_userdef.h	MIPI driver user definition Header file which defines user definition functions prototype and static constant definition

4.2 State transition

Figure 4-1 shows the state transition diagram of MIPI driver.

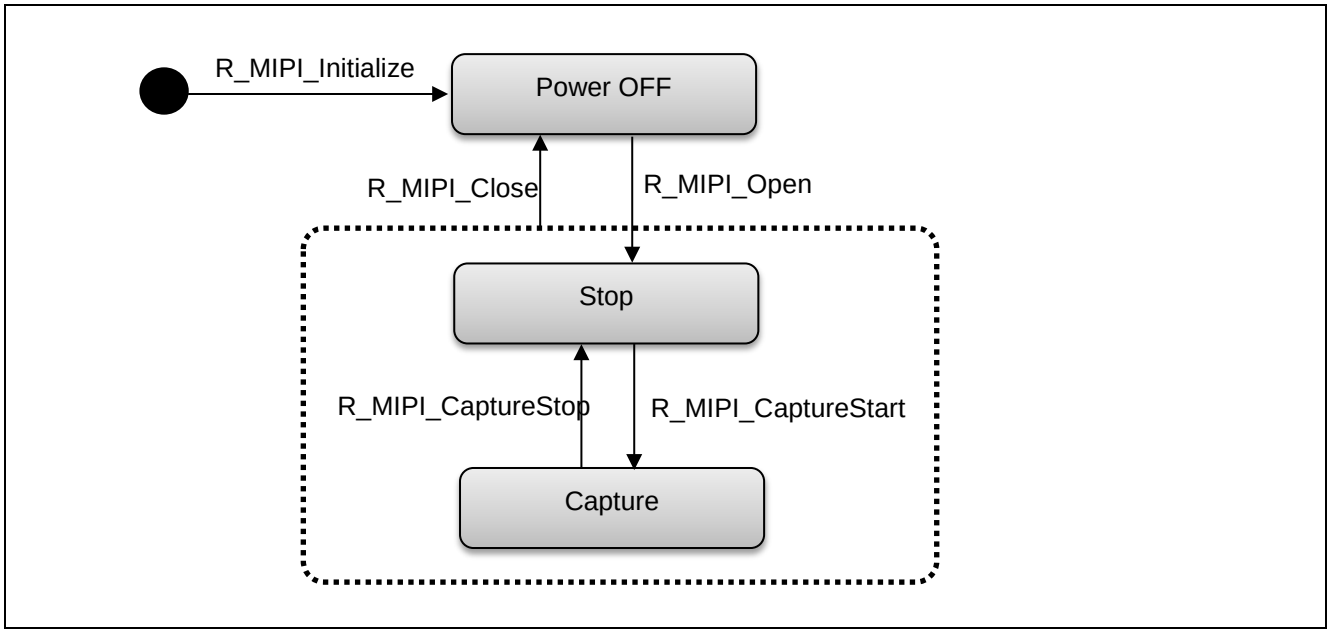

Figure 4-1 State transition diagram

4.3 Enumeration Definitions

The enumeration type definitions are given below. See section 4.4 for the error codes.

(1) e_mipi_capture_mode_t

e_mipi_capture_mode_t is an enumeration type for frame capture mode.

```
typedef enum
{
    MIPI_SINGLE_MODE = 0,
    MIPI_CONTINUOUS_MODE,
} e_mipi_capture_mode_t;
```

Enumeration constant	Description
MIPI_SINGLE_MODE	Single capture mode
MIPI_CONTINUOUS_MODE	Continuous capture mode

(2) e_mipi_inter_t

e_mipi_inter_t is an enumeration type for field detection control.

```
typedef enum
{
    MIPI_PROGRESSIVE = 0,
    MIPI_INTERLACE,
} e_mipi_inter_t;
```

Enumeration constant	Description
MIPI_PROGRESSIVE	Progressive data
MIPI_INTERLACE	Interrace data

(3) e_vin_inputformat_t

e_vin_inputformat_t is an enumeration type for input format.

```
typedef enum
{
    VIN_INPUT_YCBCR422_8 = 0,
    VIN_INPUT_YCBCR422_8I,
    VIN_INPUT_RAW8,
} e_vin_inputformat_t;
```

Enumeration constant	Description
VIN_INPUT_YCBCR422_8	YUV(=YCbCr422 8bit)
VIN_INPUT_YCBCR422_8I	UYVY
VIN_INPUT_RAW8	RAW 8bit

(4) e_vin_outputformat_t

e_vin_outputformat_t is an enumeration type for output format.

```
typedef enum
{
    VIN_OUTPUT_YCBCR422_8  = (0x00),
    VIN_OUTPUT_Y8_CbCr     = (0x02),
    VIN_OUTPUT_Y8          = (0x03),
    VIN_OUTPUT_RAW8        = (0x20),
} e_vin_outputformat_t;
```

Enumeration constant	Description
VIN_OUTPUT_YCBCR422_8	YUV (=YCbCr422 8bit)
VIN_OUTPUT_Y8_CbCr8	YC separation, YCbCr422(Y 8bit, Cb/Cr 8bit multiplexed)
VIN_OUTPUT_Y8	YC separation, Y data(8bit)
VIN_OUTPUT_RAW8	RAW 8bit

(5) e_vin_outputendian_t

e_vin_outputendian_t is an enumeration type for endian type of output format.

```
typedef enum
{
    VIN_OUUPUT_EN_LITTLE = 0,
    VIN_OUTPUT_EN_BIG,
} e_vin_outputendian_t;
```

Enumeration constant	Description
VIN_OUUPUT_EN_LITTLE	Little endian
VIN_OUTPUT_EN_BIG	Big endian

(6) e_vin_interlace_t

e_vin_interlace_t is an enumeration type for interlace mode.

```
typedef enum
{
    VIN_INTERLACE_ODD = 0,
    VIN_INTERLACE_EVEN,
    VIN_INTERLACE_BOTH,
    VIN_DINTERLACE,
    VIN_PROGRESSIVE,
} e_vin_interlace_t;
```

Enumeration constant	Description
VIN_INTERLACE_ODD	Odd-field capture mode
VIN_INTERLACE_EVEN	Even-field capture mode
VIN_INTERLACE_BOTH	Odd-/even-field capture mode
VIN_PROGRESSIVE	Progressive

(7) e_vin_input_align_t

e_vin_input_align_t is an enumeration type for data alignment in the case of input format is YCbCr422.

```
typedef enum
{
    VIN_Y_UPPER = 0,
    VIN_CB_UPPER,
} e_vin_input_align_t;
```

Enumeration constant	Description
VIN_Y_UPPER	Store Y to upper bit, CbCr to lower bit
VIN_CB_UPPER	Store CbCr to upper bit, Y to lower bit

(8) e_vin_output_swap_t

e_vin_interpolation_t is an enumeration type for byte swap. For example, set “VIN_SWAP_ON” when translating YUYV data as UYVY data.

```
typedef enum
{
    VIN_SWAP_OFF = 0,
    VIN_SWAP_ON,
} e_vin_output_swap_t;
```

Enumeration constant	Description
VIN_SWAP_OFF	Enable byte swap
VIN_SWAP_ON	Disable byte swap

(9) e_mipi_interrupt_type_t

e_mipi_interrupt_type_t is an enumeration type for interrupt factor on MIPI and VIN.

```
typedef enum
{
    MIPI_INT_LESS_THAN_WC    = 0x00000001,
    MIPI_INT_AFIFO_OF        = 0x00000002,
    MIPI_INT_VD_START        = 0x00000004,
    MIPI_INT_VD_END          = 0x00000008,
    MIPI_INT_SHP_STB         = 0x00000010,
    MIPI_INT_FSFE            = 0x00000020,
    MIPI_INT_LNP_STB         = 0x00000040,
    MIPI_INT_CRC_ERR         = 0x00000080,
    MIPI_INT_HD_WC_ZERO      = 0x00000100,
    MIPI_INT_FRM_SEQ_ERR1    = 0x00000200,
    MIPI_INT_FRM_SEQ_ERR0    = 0x00000400,
    MIPI_INT_ECC_ERR         = 0x00000800,
    MIPI_INT_ECC_CRCT_ERR    = 0x00001000,
    MIPI_INT_ULPS_START      = 0x00002000,
    MIPI_INT_ULPS_END        = 0x00004000,
    MIPI_INT_ERRSOTHS        = 0x00008000,
    MIPI_INT_ERRSOTSYNCHS    = 0x00010000,
    MIPI_INT_ERRESC          = 0x00020000,
    MIPI_INT_ERRCONTROL      = 0x00040000,
    VIN_INT_FIELD2           = 0x00100000,
    VIN_INT_VSYNC_FALL       = 0x00200000,
    VIN_INT_VSYNC_RISE       = 0x00400000,
    VIN_INT_FIELD            = 0x00800000,
    VIN_INT_SCANLINE         = 0x01000000,
    VIN_INT_FRAME            = 0x02000000,
    VIN_INT_FIFO_OF          = 0x04000000
} e_mipi_interrupt_type_t;
```

Enumeration constant	Description
MIPI_INT_LESS_THAN_WC	Length of payload data of a long packet is less than the WC value
MIPI_INT_AFIFO_OF	an overflow of the asynchronous FIFO, which stores the HS data sent from the PHY
MIPI_INT_VD_START	Start of VD output from the CSI2 (a frame start interrupt)
MIPI_INT_VD_END	End of VD output from the CSI2 (a frame end interrupt)
MIPI_INT_SHP_STB	Short packet reception interrupt
MIPI_INT_FSFE	Frame packet reception interrupt
MIPI_INT_LNP_STB	Long packet reception interrupt
MIPI_INT_CRC_ERR	CRC error interrupt
MIPI_INT_HD_WC_ZERO	WC (word count) zero interrupt
MIPI_INT_FRM_SEQ_ERR1	Frame sequence error 1 interrupt (Received an illegal Frame End packet)
MIPI_INT_FRM_SEQ_ERR0	Frame sequence error 0 interrupt (Received an illegal Frame Start packet)
MIPI_INT_ECC_ERR	ECC error interrupt
MIPI_INT_ECC_CRCT_ERR	ECC 1-bit correction interrupt
MIPI_INT_ULPS_START	Ultra-low power data transfer start interrupt
MIPI_INT_ULPS_END	Ultra-low power data transfer end interrupt
MIPI_INT_ERRSOTHS	Synchronized SOT (start of transfer) error interrupt during HS reception.
MIPI_INT_ERRSOTSYNCHS	Non-synchronizable SOT (start of transfer) error interrupt during HS reception
MIPI_INT_ERRESC	Escape mode entry error interrupt
MIPI_INT_ERRCONTROL	PHY control error interrupt
VIN_INT_FIELD2	Field interrupt
VIN_INT_VSYNC_FALL	VSYNC falling edge detect interrupt
VIN_INT_VSYNC_RISE	VSYNC rising edge detect interrupt
VIN_INT_FIELD	Field switching interrupt
VIN_INT_SCANLINE	Scanline interrupt
VIN_INT_FRAME	End of frame interrupt
VIN_INT_FIFO_OF	FIFO overflow interrupt

4.4 Error Codes

Table 4-2 shows the error code list of MIPI driver.

Table 4-2 MIPI driver error code list

Error Code	Value	Description
MIPI_OK	0	Normal termination
MIPI_STATUS_ERR	1	Status error API function is called in prohibition condition
MIPI_PARAM_ERR	2	Parameter error API function is called with illegal parameters

4.5 User Custom Parameter

Parameters that can statically be changed by the user are defined in “r_mipi_user.h” for this driver.

(1) Constant definition

Constant definitions are shown as below. For more detail of each signal timing of MIPI D-PHY, please refer the specification document of MIPI CSI-2.

Error Code	Value	Description
MIPI_INTERRUPT_PRIORITY	28u	Interrupt priority of MIPI
VIN_INTERRUPT_PRIORITY	28u	Interrupt priority of VIN
MIPI_1US_WAIT	528u	1us wait This definition is in the case of CPU clock is 528MHz

4.6 Restrictions

(1) Reentrancy

The functions of this driver are not reentrant. An unexpected driver operation may result if this driver function is called asynchronously by two or more tasks or interrupt processing rutines.

4.7 Functions

Table 4-3 shows the API function lists of MIPI driver.

Table 4-3 List of MIPI driver API functions

Function Name	Outline	Header file
R_MIPI_Initialize	Initialize processing	r_mipi_api.h
R_MIPI_Open	MIPI configuration and start-up PHY	r_mipi_api.h
R_MIPI_Close	MIPI and VIN end processing	r_mipi_api.h
R_MIPI_Setup	VIN configuration	r_mipi_api.h
R_MIPI_SetBufferAdr	Set the address of capture buffer	r_mipi_api.h
R_MIPI_InterruptEnable	Interrupt enable setup	r_mipi_api.h
R_MIPI_InterruptDisable	Interrupt disable setup	r_mipi_api.h
R_MIPI_GetInfo	Get capture information	r_mipi_api.h
R_MIPI_CaptureStart	Start capture processing	r_mipi_api.h
R_MIPI_CaptureStop	Stop capture processing	r_mipi_api.h
R_MIPI_InterruptHandler	MIPI interrupt handler	r_mipi_api.h
R_VIN_InterruptHandler	VIN interrupt handler	r_mipi_api.h
R_MIPI_CPUVAddrToSysPAddr	Convert capture buffer address processing	r_mipi_user.h
R_MIPI_OnInitialize	Sample for releasing MIPI and VIN standby state and registering the interrupt handler	r_mipi_user.h
R_MIPI_OnFinalize	Sample for setting up MIPI and VIN standby state and releasing the interrupt handler	r_mipi_user.h

5. Function Reference

5.1 R_MIPI_Initialize

R_MIPI_Initialize			
Synopsis	Initialization processing		
Header	r_mipi_api.h		
Declaration	`void R_MIPI_Initialize(void (* const init_func)(uint32_t), const uint32_t user_num);`		
Arguments	[IN]	void (* init_func)(uint32_t)	: Callback function to be registered. Specify NULL if not necessary.
	[IN]	uint32_t user_num	: Argument to the callback function Set up according to the application
Return Value	None		
Remarks	None		

(1) Description

This function initializes the MIPI driver. Since the MIPI driver will perform neither MIPI module standby release processing nor interrupt handler registration processing, it is necessary to add those processing using the callback function specified in this function. “5.14 R_MIPI_OnInitialize” is abailable as a sample function for adding those prosessing. Add the required processing while referring to that sample.

This function takes the following actions.

- Call the callback function specified in the argument.
- Initialize the internal variables to be used by this driver.

5.2 R_MIPI_Open

R_MIPI_Open			
Synopsis	MIPI configuration and start the PHY		
Header	r_mipi_api.h		
Declaration	`e_mipi_error_t R_MIPI_Open(const st_mipi_param_t * const mipi_data);`		
Arguments	[IN]	const st_mipi_param_t * const mipi_data	: Configuration Data Do not specify NULL
Return Value	MIPI_OK		: Normal termination
	MIPI_STATUS_ERR		: Driver internal status is illegal.
	MIPI_PARAM_ERR		: The mipi_data is illegal or out of range.
Remarks	This function can be called after processing R_MIPI_Initialize.		

(1) Description

This function configures the capture lane, the capture format, and the PHY settings.

This function takes following actions.

- Parameter check of configuration data.
- MIPI software reset.
- Capture setting of interlace/progressive.
- Set the virtual channel.
- Initialize the PHY
- Update the driver internal status.

(2) Parameter details

(a) st_mipi_param_t

st_mipi_param_t structure is described as below.

```
typedef struct
{
    uint8_t  mipi_lanenum;
    uint8_t  mipi_vc;
    uint8_t  mipi_interlace;
    uint8_t  mipi_laneswap;
    uint16_t mipi_frametop;
    uint16_t mipi_outputrate;
    st_mipi_phy_timing_t mipi_phy_timing;
} st_mipi_param_t;
```

Type / Member Name	Description
uint8_t mipi_lanenum	Number of transfer lane 1: 1 lane operation 2: 2 lane parallel operation
uint8_t mipi_vc	Virtual channel 0～3 Enabled virtual channel number
uint8_t mipi_interlace	Input method (T.B.D: Fixed MIPI_PROGRESSIVE at current ver.) (Note) MIPI_PROGRESSIVE: Progressive MIPI_INTERLACE: Interlace
uint8_t mipi_laneswap	Lane swapping (T.B.D: Fixed 0 at current ver.) (Note) 0：Disable lane swapping 1：Enable lane swapping
uint16_t mipi_frametop	Even field number 0x0000～0xFFFF This value is to detect top field of interlace image Set the ID of head line synchronous packet
uint16_t mipi_outputrate	MIPI transfer rate(MHz) (T.B.D: Fixed 80 at current ver.) (Note) 80～1000 Set the MIPI transfer rate
st_mipi_phy_timing_t mipi_phy_timing	PHY timing setting Set the timing of PHY data lane and clock lane Refer the “st_mipi_phy_timing_t” structure for more detail.

Note: These parameters are not supported at current driver version. Regarding each parameter, please use the fixed value which is indicated in the table.

Even-field number (mipi_frametop) is available when the input method (mipi_interlace) set as MIPI_INTERLACE.

Virtual channel (mipi_vc) means the channel which transfers data from camera.

(b) **st_mipi_phy_timing_t**

st_mipi_phy_timing_t structure is described as below.

```
typedef struct
{
    uint16_t mipi_ths_prepare;
    uint16_t mipi_ths_settle;
    uint16_t mipi_tclk_prepare;
    uint16_t mipi_tclk_settle;
    uint16_t mipi_tclk_miss;
    uint16_t mipi_t_init_slave;
} st_mipi_phy_timing_t;
```

Type / Member Name	Description
uint16_t mipi_ths_prepare	MIPI D-PHY $T_{THS\_PREPARE}$ Parameter 0x00～0x3F Setting of the duration of the LP-00 state (immediately before entry to the HS-0 state)
uint16_t mipi_ths_settle	MIPI D-PHY T_{THS_SETTLE} Parameter 0x00～0x3F Setting of the period in which a transition to the HS state is ignored after the $T_{THS\_PREPARE}$ period begins
uint16_t mipi_tclk_prepare	MIPI D-PHY $T_{CLK\_PREPARE}$ Parameter 0x00～0x3F Setting of the duration of the LP-00 state (immediately before entry to the HS-0)
uint16_t mipi_tclk_settle	MIPI D-PHY T_{CLK_SETTLE} Parameter 0x00～0x3F Setting of the period in which a transition to the HS state is ignored after the $T_{CLK\_PREPARE}$ period begins
uint16_t mipi_tclk_miss	MIPI D-PHY T_{CLK_MISS} Parameter 0x00～0x1F Setting of the period in which the absence of the clock is detected, and the HS-RX is disabled
uint16_t mipi_t_init_slave	MIPI D-PHY T_{INIT} Parameter 0x0000～0xFFFF Minimum duration of the INIT state

5.3 R_MIPI_Close

R_MIPI_Close			
Synopsis	Finalize processing of MIPI and VIN.		
Header	r_mipi_api.h		
Declaration	`e_mipi_error_t R_MIPI_Close(void (* const finalize_func)(uint32_t), const uint32_t user_num);`		
Arguments	[IN]	void (*finalize_func)(uint32_t)	: Callback function to be registered. Specify NULL if not necessary.
	[IN]	uint32_t user_num	: Argument to the callback function Set up according to the application
Return Value	MIPI_OK		: Normal termination
	MIPI_STATUS_ERR		: Driver internal status is illegal.
Remarks	This function can be called after processing R_MIPI_Open.		

(1) Description

This function stops the capture and performs MIPI software reset. Since the MIPI driver performs neither MIPI module standby configuration processing nor interrupt handler release processing, it is necessary to add those processing using the callback function specified in this function.

“5.15 R_MIPI_OnFinalize”is available as a sample function for adding those processing. Add the required processing while referring to that sample.

This function takes the following actions.

- Disable the interrupt enable registers of MIPI and VIN.
- Stop capture of VIN.
- Clear PHY settings and process the MIPI software reset.
- Clear the internal variables which are used in driver internal.
- Call the callback function specified in the argument.

5.4 R_MIPI_Setup

R_MIPI_Setup			
Synopsis	VIN configuration		
Header	r_mipi_api.h		
Declaration	`e_mipi_error_t R_MIPI_Setup(const st_vin_setup_t * const vin_setup );`		
Arguments	[IN]	const st_vin_setup_t * const vin_setup	: Configuration Data Do not specify NULL
Return Value	MIPI_OK		: Normal termination
	MIPI_STATUS_ERR		: Driver internal status is illegal.
	MIPI_PARAM_ERR		: The vin_setup is illegal or out of range.
Remarks	This function can be called after processing R_MIPI_Open Also, execute VIN setting processing while capture stop.		

(1) Description

This function sets the clipping area of capture image, input format, and stride size, and so on.

This function takes following actions.

- Check the configuration parameter
- Set various registers of VIN

(2) Parameter details

(a) st_vin_setup_t

st_vin_setup_t structure is described as below.

```
typedef struct
{
    st_vin_preclip_t   vin_preclip;
    uint8_t         vin_inputformat;
    uint8_t         vin_outputformat;
    uint8_t         vin_outputendian;
    uint8_t         vin_interlace;
    uint16_t        vin_stride;
    uint32_t        vin_ycoffset;
    e_vin_input_align_t vin_input_align;
    e_vin_output_swap_t vin_output_swap;
} st_vin_setup_t;
```

Type / Member Name	Description
st_vin_preclip_t vin_preclip	Pre-clip area Pre-clip area setting for capture image. Refer the “st_vin_preclip_t” structure for more detail.
uint8_t vin_inputformat	Input format VIN_INPUT_YCBCR422_8: YUY (=YCbCr422 8bit) VIN_INPUT_YCBCR422_8I: UYVY VIN_INPUT_RAW8: RAW 8bit
uint8_t vin_outputformat	Input format VIN_OUTPUT_YCBCR422_8: YUY (=YCbCr422 8bit) VIN_OUTPUT_Y8_CbCr: YC separation, YCbCr422(Y 8bit, Cb/Cr 8bit) VIN_OUTPUT_Y8: YC separation, Y data(8bit) VIN_OUTPUT_RAW8: RAW 8bit
uint8_t vin_outputendian	Endian type VIN_OUUPUT_EN_LITTLE：Little endian VIN_OUTPUT_EN_BIG：Big endian
uint8_t vin_interlace	Interlace mode VIN_INTERLACE_ODD: Odd-field capture mode VIN_INTERLACE_EVEN: Even-field capture mode VIN_INTERLACE_BOTH: Odd-/even-field capture mode VIN_PROGRESSIVE: Progressive capture mode
uint16_t vin_stride	Stride size of image More than 32 (Multiples of 32) Set the stride size of output image
uint32_t vin_ycoffset	UV data address offset (T.B.D: Fixed 0 at current ver.) (Note) 0～multiple of 128 Set the transfer offset address of UV data when the output format is set as YC separation.
vin_input_align_t vin_input_align	YCbCr422 input data alignment MIPI_Y_UPPER：Y in the upper bits and CbCr in the lower bits. MIPI_CB_UPPER：CbCr in the upper bits and Y in the lower bits.
vin_output_swap_t vin_output_swap	Output Data Byte Swap Mode VIN_SWAP_OFF：Bytes are not swapped in output data. VIN_SWAP_ON：Bytes are swapped in output data.

The endian type (vin_outputendian) is used when output the image data to outside memory.

Stride of image (vin_stride) should be set horizontal pre-clip size (vin_preclip_endx - vin_preclip_startx) or more. The horizontal pre-clip size is set by vin_preclip_endx and vin_preclip_startx of st_vin_preclip_t structure,

So, set the “vin_stride” that satisfy the following condition.

vin_stride >= vin_afterclip_size_x

Also, depending on the output format (vin_outputformat), it is necessary to set the parameters as follows about the stride size of image.

Output format	**Setting unit (pixel)**
VIN_OUTPUT_YCBCR422_8	64
VIN_OUTPUT_Y8_CbCr8	128
VIN_OUTPUT_Y8	128
VIN_OUTPUT_RAW8	64

The stride size of image is written to VnIS register by MIPI driver. In the case of output format is VIN_OUTPUT_RAW8, MIPI driver writes the value of the stride size of image divided by 2, to VnIS register due to hardware specification.

(b) **st_vin_preclip_t**

st_vin_preclip_t structure is described as below.

```
typedef struct
{
    uint16_t vin_preclip_starty;
    uint16_t vin_preclip_endy;
    uint16_t vin_preclip_startx;
    uint16_t vin_preclip_endx;
} st_vin_preclip_t;
```

Type / Member Name	Description
uint16_t vin_preclip_starty	Start line (vertical direction) 0～2046 (In the case of scaling: 0～2044) The value 0 means the first valid line.
uint16_t vin_preclip_endy	End line (vertical direction) 1～2047 (In the case of scaling: 3～2047)
uint16_t vin_preclip_startx	Start pixel (horizontal direction) Even value between 0 to 2042
uint16_t vin_preclip_endx	End pixel (horizontal direction) Odd value between 5 to 2047

The number of lines of vertical direction should be more than 2 lines in pre-clipped area, so, set the “vin_preclip_endy” and “vin_preclip_starty” that satisfy the following conditions.

(vin_preclip_endy - vin_preclip_starty) >= 1

In the case of vertical or horizontal scaling specified, set the “vin_preclip_endy” and “vin_preclip_starty” that satisfy the following conditions.

(vin_preclip_endy - vin_preclip_starty) >=3

The number of pixels of horizontal direction should be even value greater than 6 in pre-clipped area, so, set the “vin_preclip_endx” and “vin_preclip_startx” that satisfy the following conditions. And result of following should be odd-value.

(vin_preclip_endx - vin_preclip_startx) >=5

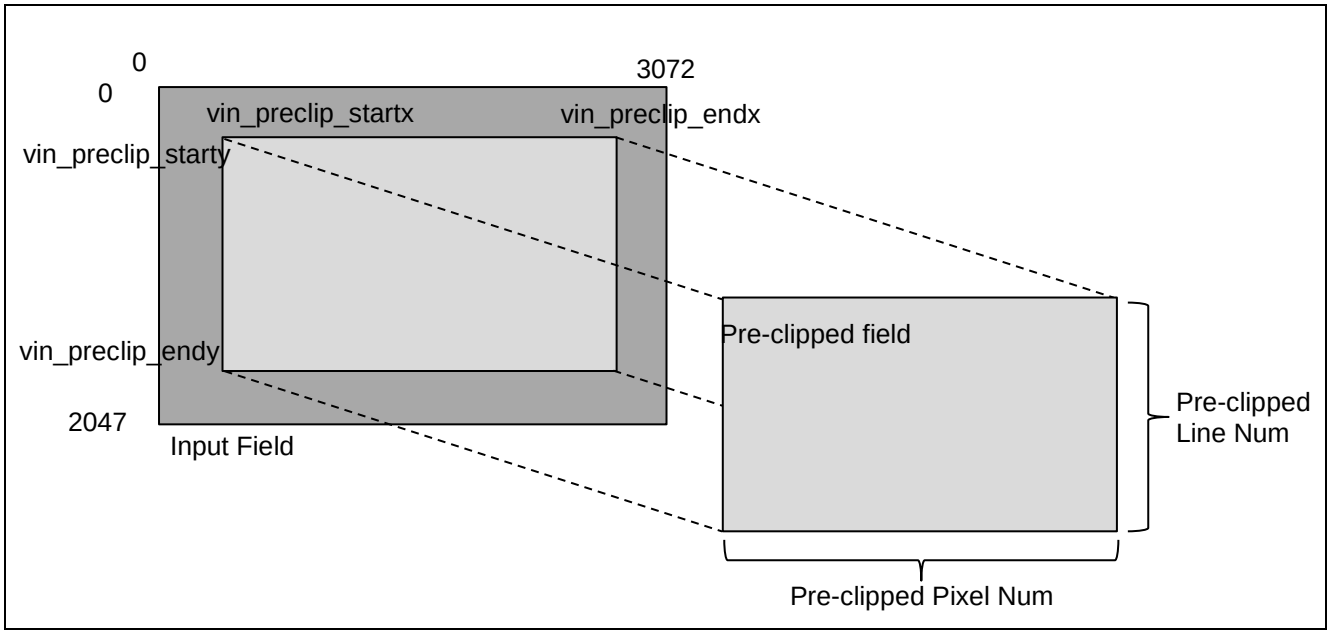

Figure 5-1 Image of pre-clipped area

5.5 R_MIPI_SetBufferAdr

R_MIPI_SetBufferAdr			
Synopsis	Set the address of capture buffer		
Header	r_mipi_api.h		
Declaration	`e_mipi_error_t R_MIPI_SetBufferAdr(const uint8_t buffer_no, const uint8_t * const bufferBase);`		
Arguments	[IN]	const uint8_t buffer_no	: Memory base number of VIN 0: Set the MB1 1: Set the MB2 2: Set the MB3
	[IN]	const uint8_t * const bufferBase	: Capture buffer address Do not specify NULL
Return Value	MIPI_OK		: Normal termination
	MIPI_STATUS_ERR		: Driver internal status is illegal.
	MIPI_PARAM_ERR		: buffer_no is out of range bufferBase is NULL, or the address is not aligned by 256bytes
Remarks	This function can be called after processing R_MIPI_Open		

(1) Description

Assign the specified buffer address (bufferBase) to memory register of VIN. VIN has 3 memory base register such as MB1, MB2, and MB3. This function stores the address which is specified by second argument to memory base register which specified by first arguments. The address which is specified by second argument should be aligned by 256bytes.

The capture sequence is MB1 -> MB2-> MB3 in the case of continuous capture mode. In the case of single capture mode, the capture data is stored to the address which is stored to MB1.

This function takes following actions.

- Check the driver internal status.
- Store the address to memory base register.

5.6 R_MIPI_InterruptEnable

R_MIPI_InterruptEnable			
Synopsis	Interrupt enable setup		
Header	r_mipi_api.h		
Declaration	`void R_MIPI_InterruptEnable(const st_mipi_int_t * const param );`		
Arguments	[IN]	const st_mipi_int_t * const param	: Interrupt settings Do not specify NULL
Return Value	None		
Remarks	None		

(1) Description

This function enables interrupts of MIPI and VIN according to interrupt settings argument.

This function takes following action

- Enable interrupts of MIPI and VIN

(2) Parameter Details

(a) st_mipi_int_t

st_mipi_int_t structure is described as below.

```
typedef struct
{
    e_mipi_interrupt_type_t type;
    void (* p_mipiCallback) (e_mipi_interrupt_type_t interrupt_flag);
    void (* p_vinCallback)  (e_mipi_interrupt_type_t interrupt_flag);
    uint32_t line_num;
} st_mipi_int_t;
```

Type / Member Name	Description
e_mipi_interrupt_type_t type	Interrupt factor of MIPI and VIN Choice the interrupt factor needed which defined at e_mipi_interrupt_type_t of “4.3 Enumeration Definitions”
void (* p_mipiCallback) (e_mipi_interrupt_type_t interrupt_flag)	MIPI interrupt callback function Callback function which is called when MIPI interrupt occurs. Do not specify NULL.
void (* p_vinCallback) (e_mipi_interrupt_type_t interrupt_flag)	VIN Interrupt callback function Callback function which is called when VIN interrupt occurs. Do not specify NULL.
uint32_t line_num	Line number for scan line interrupt 0x0000～0x07FF Set the line number in the case of type is VIN_INT_SCANLINE.

5.7 R_MIPI_InterruptDisable

R_MIPI_InterruptDisable	
Synopsis	Interrupt disable setup
Header	r_mipi_api.h
Declaration	`void R_MIPI_InterruptDisable(void);`
Arguments	None
Return Value	None
Remarks	None

(1) Description

Disable interrupts of MIPI and VIN. Also this function clears the interrupt callback registration of MIPI and VIN.

This function takes following action.

- Disable interrupt of MIPI and VIN

5.8 R_MIPI_GetInfo

R_MIPI_GetInfo			
Synopsis	Get capture information		
Header	r_mipi_api.h		
Declaration	`e_mipi_error_t R_MIPI_GetInfo(st_vin_info_type_t * infoType);`		
Arguments	[IN]	st_vin_info_type_t * infoType	: Address to store capture information Do not specify NULL
Return Value	MIPI_OK		: Normal termination
	MIPI_STATUS_ERR		: Driver internal status is illegal.
Remarks	This function can be called after processing R_MIPI_Open This function returns error when call this function while capturing.		

(1) Description

Store the capture information, such as current capture field, place of capture line, and valid frame buffer (memory base register), to specified address.

This function takes following actions.

- Check driver internal status.
- Store the capture information to specified address.

(2) Parameter details

(a) st_vin_info_type_t
st_vin_info_type_t structure is described as below.

```
typedef struct
{
    uint16_t vin_nowcaptureline;
    uint8_t vin_nowcapturefield;
    uint8_t vin_nowcapturebase;
} st_vin_info_type_t;
```

Type / Member Name	Description
uint16_t vin_nowcaptureline	Line count The line place of current capture field
uint8_t vin_nowcapturefield	Current capture field 0: Odd-field 1: Even-field
uint8_t vin_nowcapturebase	Valid frame buffer 0: Valid frame buffer is MB1 1: Valid frame buffer is MB2 2: Valid frame buffer is MB3 3: There is no valid frame buffer.

5.9 R_MIPI_CaptureStart

R_MIPI_CaptureStart			
Synopsis	Start capture processing		
Header	r_mipi_api.h		
Declaration	`e_mipi_error_t R_MIPI_CaptureStart(const e_mipi_capture_mode_t captureMode);`		
Arguments	[IN]	const e_mipi_capture_mode_t captureMode	: Capture mode MIPI_SINGLE_MODE: Single capture MIPI_CONTINUOUS_MODE: Continuous capture
Return Value	MIPI_OK		: Normal termination
	MIPI_STATUS_ERR		: Driver internal status is illegal.
Remarks	This function can be called after processing R_MIPI_Open Set several capture settings by R_MIPI_Setup and R_MIPI_SetBufferAdr before start capture by calling this function.		

(1) Description

Start capture processing after set capture mode.

This function takes following actions

- Check driver internal status
- Set capture mode
- Start capture

5.10 R_MIPI_CaptureStop

R_MIPI_CaptureStop		
Synopsis	Stop capture processing	
Header	r_mipi_api.h	
Declaration	`e_mipi_error_t R_MIPI_CaptureStop(void);`	
Arguments	None	
Return Value	MIPI_OK	: Normal termination
	MIPI_STATUS_ERR	: Driver internal status is illegal.
Remarks	This function can be called while capture processing by starting R_MIPI_CaptureStart	

(1) Description

Stop capture processing.

This function takes following actions.

- Check driver internal status
- Stop capture

5.11 R_MIPI_InterruptHandler

R_MIPI_InterruptHandler			
Synopsis	MIPI interrupt handler		
Header	r_mipi_api.h		
Declaration	`void R_MIPI_InterruptHandler( uint32_t int_sense );`		
Arguments	[IN]	uint32_t int_sense	: Interrupt request edge/level
Return Value	None		
Remarks	None		

(1) Description

This function is MIPI interrupt handler. This function is registered as the MIPI interrupt handler through the function described in section “5.14 R_MIPI_OnInitialize” which is introduced as an example of interrupt handler registration processing.

5.12 R_VIN_InterruptHandler

R_VIN_InterruptHandler			
Synopsis	VIN interrupt handler		
Header	r_mipi_api.h		
Declaration	`void R_VIN_InterruptHandler( uint32_t int_sense );`		
Arguments	[IN]	uint32_t int_sense	: Interrupt request edge/level
Return Value	None		
Remarks	None		

(1) Description

This function is MIPI interrupt handler. This function is registered as the MIPI interrupt handler through the function described in section “5.15 R_MIPI_OnFinalize” which is introduced as an example of interrupt handler registration processing.

5.13 R_MIPI_CPUVAddrToSysPAddr

R_MIPI_CPUVAddrToSysPAddr			
Synopsis	Convert capture buffer address processing		
Header	r_mipi_user.h		
Declaration	`uint32_t R_MIPI_CPUVAddrToSysPAddr(uint32_t vaddr);`		
Arguments	[IN]	uint32_t vaddr	: Virtual address
Return Value	Integer of uint32_t type		: Physical address
Remarks	None		

(1) Description

This function converts the virtual address to physical address. This function is called from R_MIPI_SetBufferAdr when the R_MIPI_SetBufferAdr processing store the address to memory base register.

5.14 R_MIPI_OnInitialize

R_MIPI_OnInitialize	
Synopsis	Sample for releasing MIPI and VIN standby state and registering the interrupt handler
Header	r_mipi_user.h
Declaration	`void R_MIPI_OnInitialize (const uint32_t user_num);`
Arguments	[IN] const uint32_t user_num : User parameter
Return Value	None
Remarks	None

(1) Description

This function is prepared as an example of releasing the MIPI and VIN module standby state and registering an interrupt handler. Please implement processing which user environment required.

This function takes following actions.

- Release standby of MIPI and VIN
- Register an interrupt handler
- Set up interrupt priorities

5.15 R_MIPI_OnFinalize

R_MIPI_OnFinalize	
Synopsis	Sample for setting up MIPI and VIN standby state and releasing the interrupt handler
Header	r_mipi_user.h
Declaration	`void R_MIPI_OnFinalize( const uint32_t user_num );`
Arguments	[IN] const uint32_t user_num : User parameter
Return Value	None
Remarks	None

(1) Description

This function is prepared as an example of setting up the MIPI and VIN module standby state and releasing the interrupt handler. Please implement processing which user environment required.

This function takes following actions.

- Set up MIPI and VIN standby
- Release the interrupt handler

6. How to Import the Driver

6.1 e² studio

Please refer to the RZ/A2M Smart Configurator User's Guide: e² studio R20AN0583EJ for details on how to import drivers into projects in e2 studio using the Smart Configurator tool.

6.2 For Projects created outside e² studio

This section describes how to import the driver into your project.

Generally, there are two steps in any IDE:

1) Copy the driver to the location in the source tree that you require for your project.

2) Add the link to where you copied your driver to the compiler.

Other required drivers, e.g. r_cbuffer, must be imported similarly.

7. Reference Documents

User's Manual: Hardware

RZ/A2M Group User's Manual: Hardware

The latest version can be downloaded from the Renesas Electronics website.

RTK7921053C00000BE (RZ/A2M CPU board) User's Manual

The latest version can be downloaded from the Renesas Electronics website.

RTK79210XXB00000BE (RZ/A2M SUB board) User's Manual

The latest version can be downloaded from the Renesas Electronics website.

ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition Issue C

The latest version can be downloaded from the ARM website.

ARM Cortex™-A9 (Revision: r4p1) Technical Reference Manual

The latest version can be downloaded from the ARM website.

ARM Generic Interrupt Controller Architecture Specification - Architecture version 2.0

The latest version can be downloaded from the ARM website.

ARM CoreLink™ Level 2 Cache Controller L2C-310 (Revision: r3p3) Technical Reference Manual

The latest version can be downloaded from the ARM website.

Technical Update/Technical News

The latest information can be downloaded from the Renesas Electronics website.

User's Manual: Development Tools

Integrated development environment e2studio User’s Manual can be downloaded from the Renesas Electronics website.

The latest version can be downloaded from the Renesas Electronics website.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Sep.14, 2018	-	First edition issued
1.01	Dec. 28, 2018	21	Additional parameter of R_MIPI_Setup.
1.10	May. 17, 2019	3	Table 2.1 Peripheral device used(1/2) Remove compiler option "-mthumb-interwork"
		35	Added “6.How to Import the Driver”

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.